**sql四个排名函数 生成记录序号**

排名函数是SQL Server2005新加的功能。在SQL Server2005中有如下四个排名函数：  
  
1. **row\_number**   顺序生成序号

2. **rank**       相同的序值序号相同，但序号会跳号

3. **dense\_rank**   相同的序值序号相同，序号顺充递增

4. **ntile**     装桶，把记录分成指的桶数，编序号  
    下面分别介绍一下这四个排名函数的功能及用法。在介绍之前假设有一个t\_table表，表结构与表中的数据如图1所示：

  
  
图1

其中field1字段的类型是int，field2字段的类型是varchar

一、row\_number

    row\_number函数的用途是非常广泛，这个函数的功能是为查询出来的每一行记录生成一个序号。row\_number函数的用法如下面的SQL语句所示：

select row\_number() over(order by field1) as row\_number,\* from t\_table

    上面的SQL语句的查询结果如图2所示。

  
  
图2

    其中row\_number列是由row\_number函数生成的序号列。在使用row\_number函数是要使用over子句选择对某一列进行排序，然后才能生成序号。

    实际上，row\_number函数生成序号的基本原理是先使用over子句中的排序语句对记录进行排序，然后按着这个顺序生成序号。over子句中的order by子句与SQL语句中的order by子句没有任何关系，这两处的order by 可以完全不同，如下面的SQL语句所示：

select row\_number() over(order by field2 desc) as row\_number,\* from t\_table order by field1 desc

    上面的SQL语句的查询结果如图3所示。

  
  
图3

    我们可以使用row\_number函数来实现查询表中指定范围的记录，一般将其应用到Web应用程序的分页功能上。下面的SQL语句可以查询t\_table表中第2条和第3条记录：

with t\_rowtable  
as  
(  
    select row\_number() over(order by field1) as row\_number,\* from t\_table  
)  
select \* from t\_rowtable where row\_number>1 and row\_number < 4 order by field1

    上面的SQL语句的查询结果如图4所示。

  
  
图4

    上面的SQL语句使用了CTE，关于CTE的介绍将读者参阅[《SQL Server2005杂谈（1）：使用公用表表达式（CTE）简化嵌套SQL》](http://www.blogjava.net/nokiaguy/archive/2009/nokiaguy/archive/2009/01/31/252787.html)。  
    另外要注意的是，如果将row\_number函数用于分页处理，over子句中的order by 与排序记录的order by 应相同，否则生成的序号可能不是有续的。  
    当然，不使用row\_number函数也可以实现查询指定范围的记录，就是比较麻烦。一般的方法是使用颠倒Top来实现，例如，查询t\_table表中第2条和第3条记录，可以先查出前3条记录，然后将查询出来的这三条记录按倒序排序，再取前2条记录，最后再将查出来的这2条记录再按倒序排序，就是最终结果。SQL语句如下：

select \* from (select top 2 \* from( select top 3 \* from t\_table order by field1) a order by field1 desc) b order by field1

    上面的SQL语句查询出来的结果如图5所示。  
  
  
图5

    这个查询结果除了没有序号列row\_number，其他的与图4所示的查询结果完全一样。

二、rank

    rank函数考虑到了over子句中排序字段值相同的情况，为了更容易说明问题，在t\_table表中再加一条记录，如图6所示。  
  
  
图6

    在图6所示的记录中后三条记录的field1字段值是相同的。如果使用rank函数来生成序号，这3条记录的序号是相同的，而第4条记录会根据当前的记录数生成序号，后面的记录依此类推，也就是说，在这个例子中，第4条记录的序号是4，而不是2。rank函数的使用方法与row\_number函数完全相同，SQL语句如下：

select rank() over(order by field1),\* from t\_table order by field1

    上面的SQL语句的查询结果如图7所示。  
  
  
图7

三、dense\_rank

    dense\_rank函数的功能与rank函数类似，只是在生成序号时是连续的，而rank函数生成的序号有可能不连续。如上面的例子中如果使用dense\_rank函数，第4条记录的序号应该是2，而不是4。如下面的SQL语句所示：

select dense\_rank() over(order by field1),\* from t\_table order by field1

    上面的SQL语句的查询结果如图8所示。

  
  
图8

    读者可以比较图7和图8所示的查询结果有什么不同

四、ntile  
    ntile函数可以对序号进行分组处理。这就相当于将查询出来的记录集放到指定长度的数组中，每一个数组元素存放一定数量的记录。ntile函数为每条记录生成的序号就是这条记录所有的数组元素的索引（从1开始）。也可以将每一个分配记录的数组元素称为“桶”。ntile函数有一个参数，用来指定桶数。下面的SQL语句使用ntile函数对t\_table表进行了装桶处理：

select ntile(4) over(order by field1) as bucket,\* from t\_table

    上面的SQL语句的查询结果如图9所示。  
  
  
图9

    由于t\_table表的记录总数是6，而上面的SQL语句中的ntile函数指定了桶数为4。

    也许有的读者会问这么一个问题，SQL Server2005怎么来决定某一桶应该放多少记录呢？可能t\_table表中的记录数有些少，那么我们假设t\_table表中有59条记录，而桶数是5，那么每一桶应放多少记录呢？

    实际上通过两个约定就可以产生一个算法来决定哪一个桶应放多少记录，这两个约定如下：

1. 编号小的桶放的记录不能小于编号大的桶。也就是说，第1捅中的记录数只能大于等于第2桶及以后的各桶中的记录。

2. 所有桶中的记录要么都相同，要么从某一个记录较少的桶开始后面所有捅的记录数都与该桶的记录数相同。也就是说，如果有个桶，前三桶的记录数都是10，而第4捅的记录数是6，那么第5桶和第6桶的记录数也必须是6。

    根据上面的两个约定，可以得出如下的算法：

    // mod表示取余，div表示取整   
    if(记录总数 mod 桶数 == 0)  
    {  
        recordCount = 记录总数 div 桶数;  
        将每桶的记录数都设为recordCount  
    }   
    else  
    {  
        recordCount1 = 记录总数 div 桶数 + 1;  
        int n = 1;  //  n表示桶中记录数为recordCount1的最大桶数  
        m = recordCount1 \* n;  
        while(((记录总数 - m)  mod  (桶数 -  n))  != 0 )  
        {  
            n++;  
            m = recordCount1 \* n;  
        }   
        recordCount2 = (记录总数 - m) div  (桶数 - n);  
        将前n个桶的记录数设为recordCount1  
        将n + 1个至后面所有桶的记录数设为recordCount2  
    }

    根据上面的算法，如果记录总数为59，桶数为5，则前4个桶的记录数都是12，最后一个桶的记录数是11。

    如果记录总数为53，桶数为5，则前3个桶的记录数为11，后2个桶的记录数为10。

    就拿本例来说，记录总数为6，桶数为4，则会算出recordCount1的值为2，在结束while循环后，会算出recordCount2的值是1，因此，前2个桶的记录是2，后2个桶的记录是1。